AMENDMENTS TO THE CLAIMS

1. (Currently Amended) A method in a computer system for accessing a collection of data items, the method comprising:

when adding a data item to the collection,

fetching and adding to a write counter, the fetched write counter pointing to a bucket within a bucket array;

reading from the bucket pointed to by the fetched write ~~pointer~~ counter using a synchronization access mode of sync so that if the bucket is currently marked empty, the reading is delayed until the bucket is marked full and so that the bucket is marked empty upon reading to prevent subsequent reading from the bucket until the bucket is written to;

after reading the bucket, storing the data item in association with the bucket pointed to by the fetched write ~~pointer~~counter;

writing to the bucket pointed to by the fetched write ~~pointer~~ counter using a synchronization access mode of sync so that the bucket is marked full allowing reading from the bucket; and

fetching and adding to a lower bound to indicate ~~the number of~~that the data ~~items~~ item has been added to the collection.

2. (Currently Amended) The method of claim 1 wherein the bucket pointed to by the fetched write ~~pointer~~ counter contains a pointer to a linked list of data items.

3. (Currently Amended) The method of claim 1 wherein the fetched write ~~pointer~~ counter modulo a number of buckets in the bucket array points to a bucket within the bucket array.

4. (Original) The method of claim 1 wherein the adding adds one to the write counter.

5. (Original) The method of claim 1 wherein the adding adds a size of a bucket to the write counter.

6. (Currently Amended) The method of claim 1 including

when removing a data item from the collection,

fetching and adding to a read counter, the fetched read counter pointing to a bucket within the bucket array;

reading from the bucket pointed to by the fetched read ~~pointer~~ counter using a synchronization access mode of sync;

removing the data item from association with the bucket pointed to by the fetched read ~~pointer~~counter; and

writing to the bucket pointed to by the fetched ~~write pointer~~read counter using a synchronization access mode of sync.

7. (Currently Amended) The method of claim 1 including

when removing a data item from the collection,

checking the lower bound to ensure that the collection contains a data item, when it cannot be ensured that the collection contains a data item, indicating that the collection may be empty; and

when it can be ensured that the collection contains a data item,

fetching and adding to a read counter, the fetched read counter pointing to a bucket within the bucket array;

reading from the bucket pointed to by the fetched read ~~pointer~~ counter using a synchronization access mode of sync;

removing the data item from association with the bucket pointed to by the fetched read ~~pointer~~counter; and

writing to the bucket pointed to by the fetched ~~write pointer~~read counter using a synchronization access mode of sync.

8. (Original) The method of claim 7 wherein the checking includes fetching and adding a negative number to the lower bound.

9. (Original) The method of claim 8 wherein the checking includes fetching and adding a positive number to the lower bound when it cannot be ensured that the collection contains an item.

10. (Original) The method of claim 1 wherein the synchronization access mode of sync prevents simultaneous access of a bucket by multiple threads.

11. (Original) The method of claim 1 wherein the collection of data items is stored in an array data structure.

12. (Original) The method of claim 1 wherein the collection of data items is stored in a linked list data structure.

13. (Original) The method of claim 1 wherein the collection of data items is stored in a tree data structure.

14-15. (Cancelled)

16. (Original) The method of claim 1 wherein the writing is permitted only when the bucket is empty.

17. (Cancelled)

18. (Currently Amended) A method in a computer system for accessing a collection of data items, the method comprising:

when removing a data item from the collection,

fetching and adding to a read counter, the fetched read counter pointing to a bucket within the bucket array;

reading from the bucket pointed to by the fetched read ~~pointer~~ counter using a synchronization access mode of sync so that if the bucket is currently marked empty, the reading is delayed until the bucket is marked full and so that the bucket is marked empty upon reading to prevent subsequent reading from the bucket until the bucket is written to;

removing the data item from association with the bucket pointed to by the fetched read ~~pointer~~counter; and

writing to the bucket pointed to by the fetched ~~write pointer~~read counter using a synchronization access mode of sync so that the bucket is marked full allowing reading from the bucket.

19. (Original) The method of claim 18 including before fetching and adding to the read counter checking a lower bound to ensure that the collection contains a data item.

20. (Original) The method of claim 19 wherein it cannot be ensured that the collection contains a data item, indicating that a data item cannot be removed.

21. (Currently Amended) The method of claim 18 wherein the bucket pointed to by the fetched read ~~pointer~~ counter contains a pointer to a linked list of data items.

22. (Currently Amended) The method of claim 18 wherein the fetched read ~~pointer~~ counter modulo a number of buckets in the bucket array points to a bucket within the bucket array.

23. (Original) The method of claim 18 wherein the removing adds one to the read counter.

24. (Original) The method of claim 18 wherein the adding adds a size of a bucket to the read counter.

25. (Original) The method of claim 18 wherein the synchronization access mode of sync prevents simultaneous access of a bucket by multiple threads.

26. (Original) The method of claim 18 wherein the collection of data items is stored in an array data structure.

27. (Original) The method of claim 18 wherein the collection of data items is stored in a linked list data structure.

28. (Original) The method of claim 18 wherein the collection of data items is stored in a tree data structure.

29-30. (Cancelled)

31. (Original) The method of claim 18 wherein the writing is permitted only when the bucket is empty.

32. (Cancelled)

33. (Currently Amended) A method in a computer system for accessing a collection of data items, the method comprising:

defining a pointer to indicate a location for a ~~data item~~bucket;

defining a lower bound to indicate a number of items in the collection; and

determining based on the lower bound whether the collection has a data item; and

when it is determined that the collection has a data item,

adjusting the lower bound to indicate that the collection has one less data item; ~~and~~

~~accessing~~ reading from the ~~data item~~bucket at the location defined by the pointer using ~~an~~ a synchronization access mode of sync so that if the bucket is currently marked empty, the reading is delayed until the bucket is marked full and so that the bucket is marked empty upon reading to prevent subsequent reading from the bucket until the bucket is written to;

removing a data item from association with the bucket at the location defined by the pointer; and

writing to the bucket at the location defined by the pointer using a synchronization access mode of sync so that the bucket is marked full allowing reading from the bucket.

34-35. (Cancelled)

36. (Original) The method of claim 33 wherein write access to the location is permitted only when the location is empty.

37. (Cancelled)

38. (Original) The method of claim 33 wherein the data items of the collection are accessed by multiple readers and writers.

39. (Original) The method of claim 33 wherein the data items of the collection are accessed by multiple producers.

40. (Original) The method of claim 33 wherein the data items of the collection are accessed by multiple consumers.

41. (Original) The method of claim 33 including

when access to the location by a thread is blocked,

enabling an exception to be raised when the location is next accessed; and

blocking the thread; and

when an exception is raised as a result of access by another thread to that location,

completing the access by that other thread to that location; and

restarting execution of the blocked thread.

42. (Original) The method of claim 41 wherein when access by the thread to the location is blocked, saving a state of the thread and storing a reference to the thread in the location.

43. (Original) The method of claim 42 wherein the reference is a pointer to a data structure that identifies the blocked thread and the saved state.

44. (Original) The method of claim 43 wherein the data structure indicates the value that was stored in the location before storing the reference.

45. (Original) The method of claim 33 wherein the collection includes an array of buckets, each bucket including a pointer to data items.

46. (Currently Amended) The method of claim 45 wherein the collection includes a write ~~pointer~~ counter that indicates a bucket into which a next data item is to be stored and a read ~~pointer~~ counter that indicates a bucket from which a next data item is to be read.

47. (Original) The method of claim 45 wherein the pointed to data items are stored in a linked list.

48. (Original) The method of claim 45 wherein multiple readers and writers can be accessing data items of different buckets simultaneously.

49. (Currently Amended) A computer system for accessing data, comprising:

a collection of buckets;

a read counter and a write counter that point to buckets within the collection;

a data structure for each bucket with locations for holding data at each of the buckets; ~~and~~

a lower bound indicating whether a data item is currently stored in the data structure;

a component that adjusts the lower bound to indicate a change in the number of data items;

a component that accesses the bucket at the location defined by the pointer using a synchronization access mode of sync so that if the bucket is currently marked empty, the reading is delayed until the bucket is marked full and so that the bucket is marked empty upon reading to prevent subsequent reading from the bucket until the bucket is written to;

a component that modifies data items associated with the bucket at the location defined by the pointer; and

a component that writes to the bucket at the location defined by the pointer using a synchronization access mode of sync so that the bucket is marked full allowing reading from the bucket.

50. (Cancelled)

51. (Currently Amended) The computer system of claim ~~45~~ 49 wherein write access to a bucket is permitted only when the bucket is empty.

52. (Currently Amended) The computer system of claim ~~45~~ 49 including accessing programs that each operates in a different thread.

53. (Currently Amended) The computer system of claim ~~45~~ 49 wherein the data is accessed by multiple reading threads and writing threads.

54. (Currently Amended) The computer system of claim ~~45~~ 49 including

when access by a thread to a bucket is blocked,

enabling an exception to be raised when the bucket is next accessed; and

blocking the thread; and

when an exception is raised as a result of access by another thread to that bucket,

completing the access by that other thread to that bucket; and

restarting execution of the blocked thread.

55-59. (Cancelled)